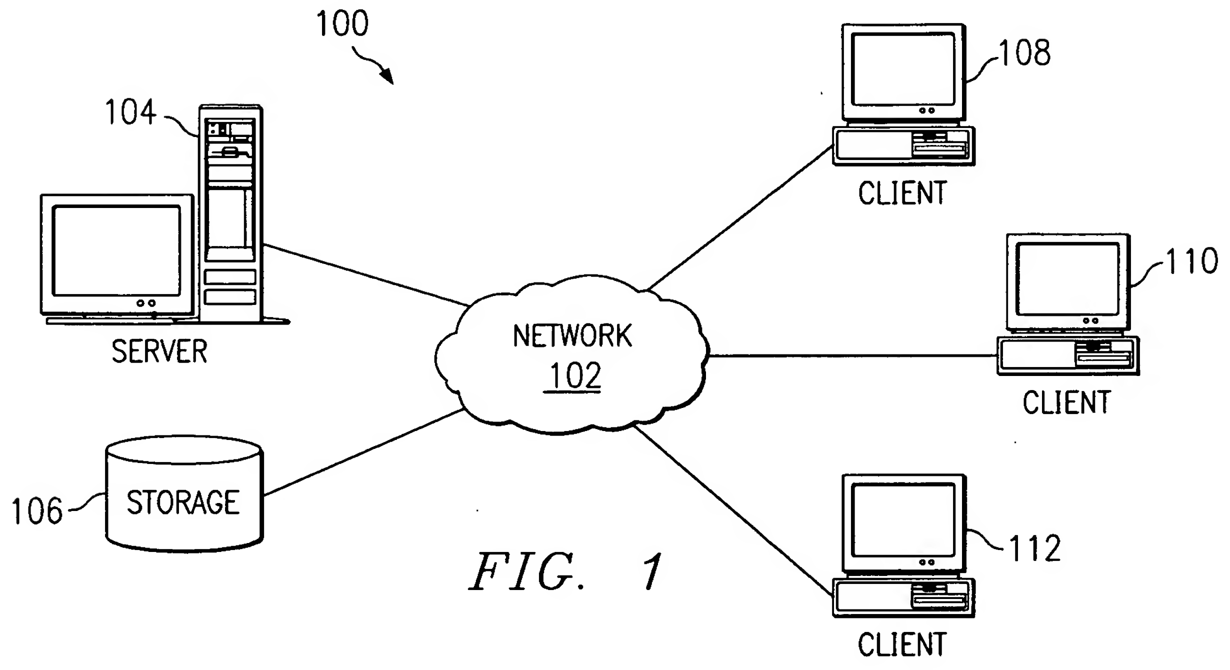

100
104
SERVER
106
STORAGE
NETWORK
102
108
CLIENT
110
CLIENT
112
CLIENT

FIG. 1

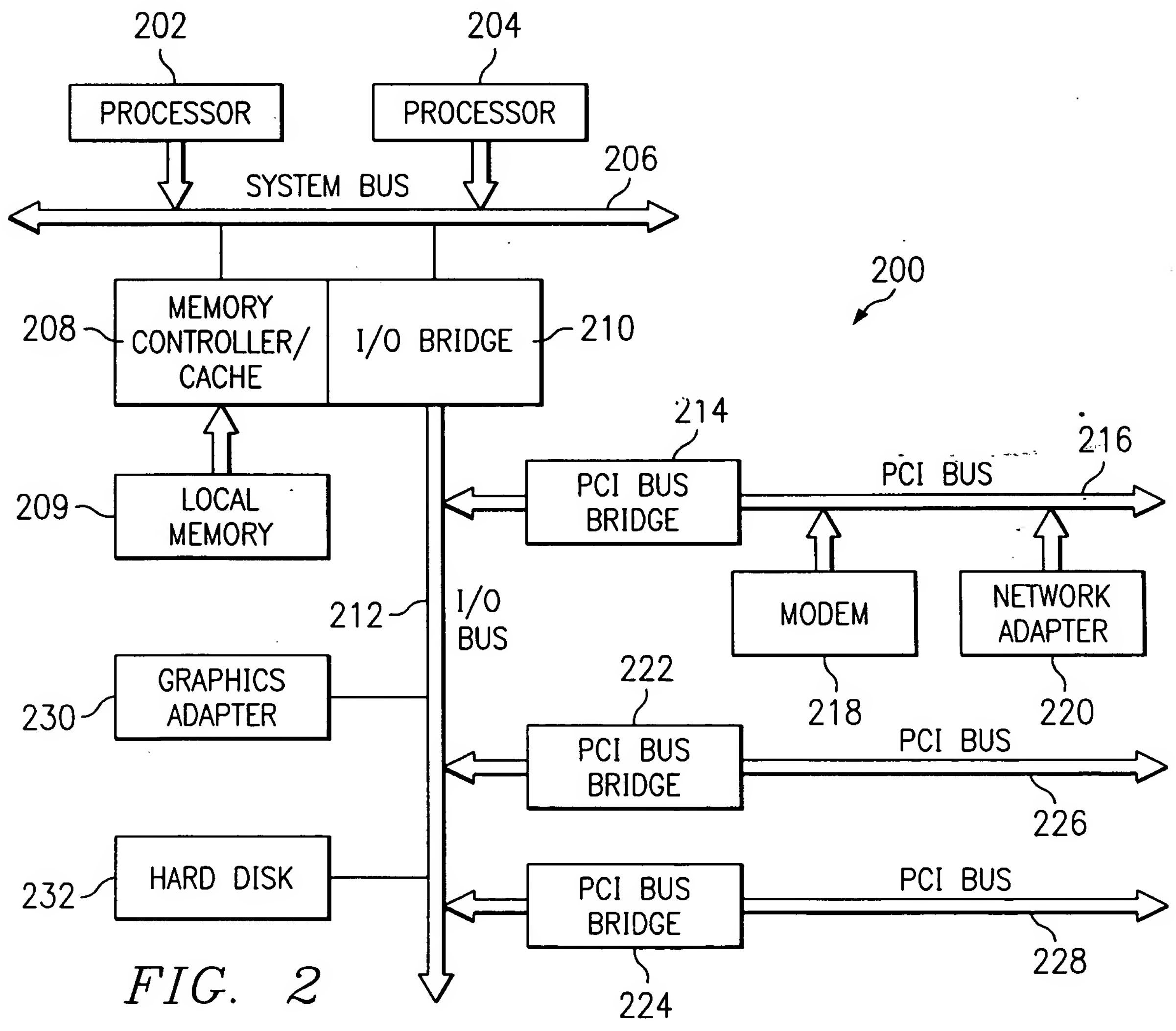

202
PROCESSOR
204
PROCESSOR
SYSTEM BUS
206
200
208
MEMORY CONTROLLER/ CACHE
I/O BRIDGE
210
209
LOCAL MEMORY
214
PCI BUS BRIDGE
PCI BUS
216
MODEM
218
NETWORK ADAPTER
220
212
I/O BUS
222
PCI BUS BRIDGE
PCI BUS
226
230
GRAPHICS ADAPTER
232
HARD DISK
PCI BUS BRIDGE
PCI BUS
228
224

FIG. 2

FIG. 3

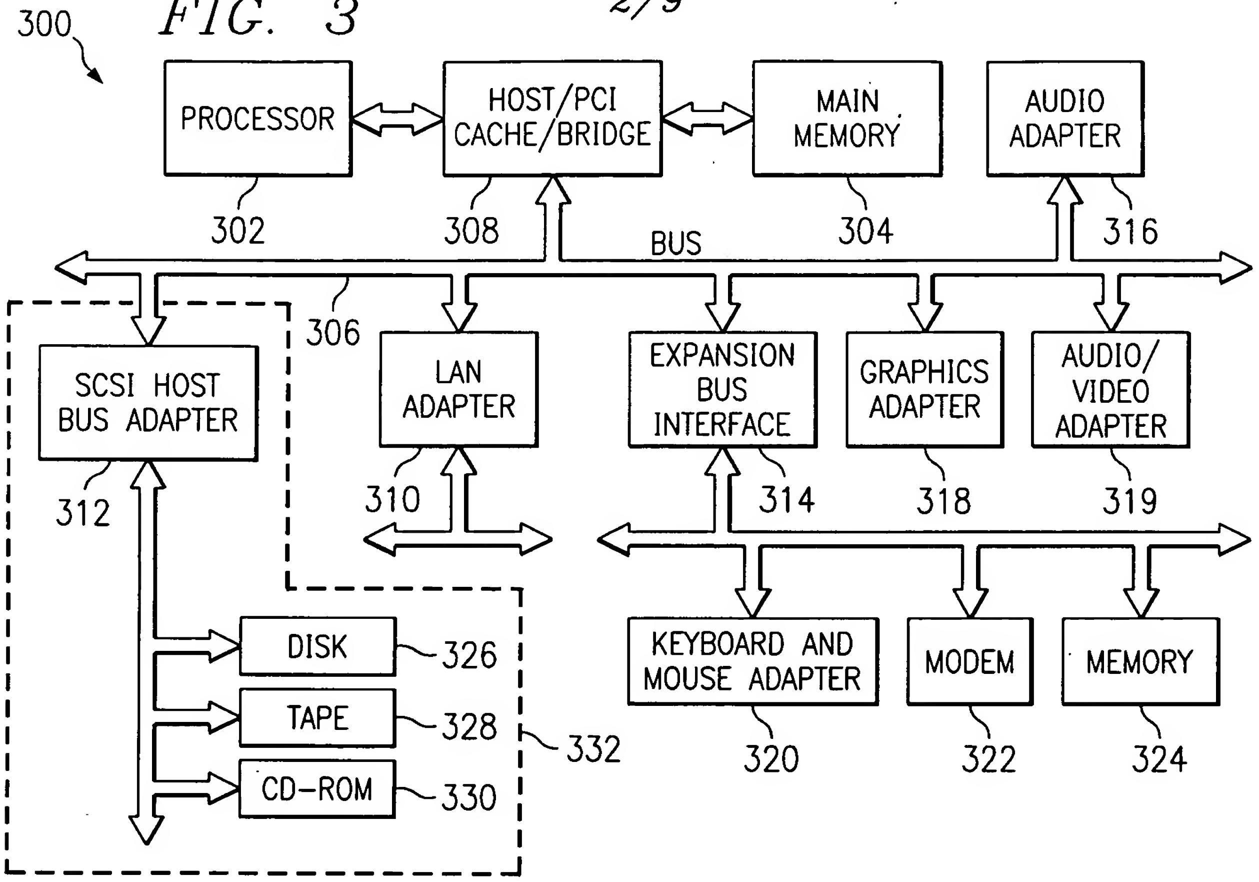
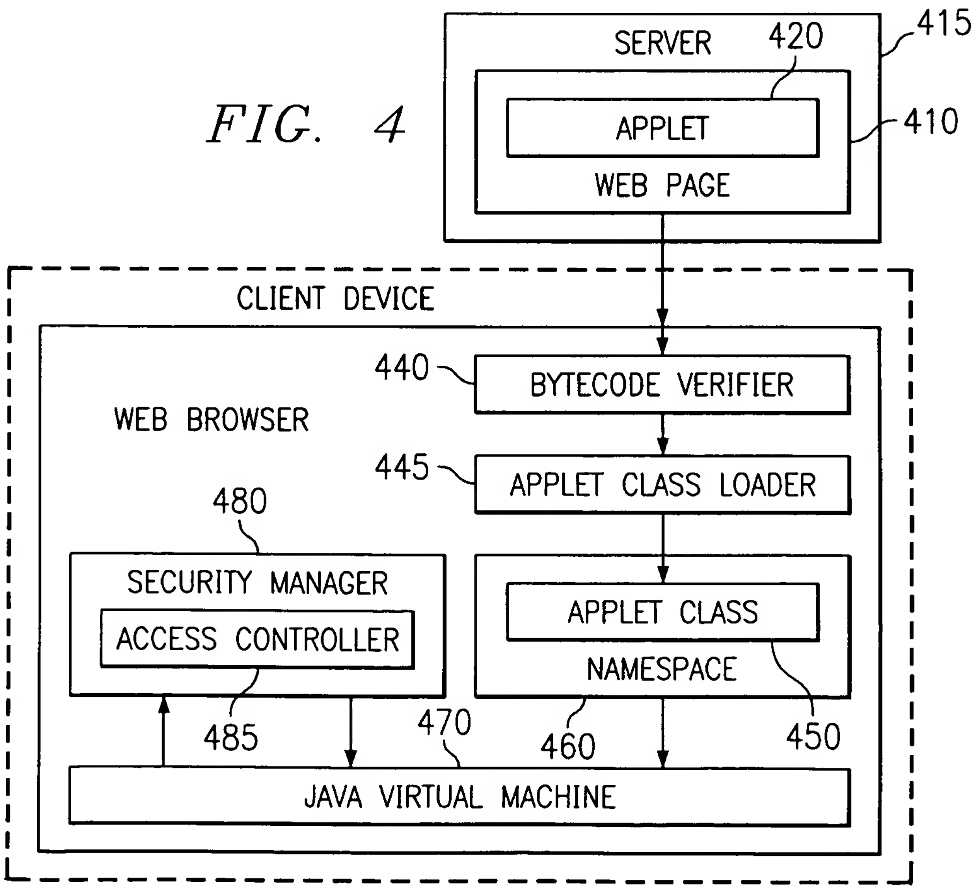

300
PROCESSOR
302
HOST/PCI CACHE/BRIDGE
308
MAIN MEMORY
304
AUDIO ADAPTER
316
BUS
306
SCSI HOST BUS ADAPTER
312
LAN ADAPTER
310
EXPANSION BUS INTERFACE
314
GRAPHICS ADAPTER
318
AUDIO/ VIDEO ADAPTER
319
DISK
326
TAPE
328
CD-ROM
330
332
KEYBOARD AND MOUSE ADAPTER
320
MODEM
322
MEMORY
324

FIG. 4


SERVER
420
415
APPLET
410
WEB PAGE
CLIENT DEVICE
440
BYTECODE VERIFIER
WEB BROWSER
445
APPLET CLASS LOADER
480
SECURITY MANAGER
ACCESS CONTROLLER
APPLET CLASS
NAMESPACE
485
470
460
450
JAVA VIRTUAL MACHINE

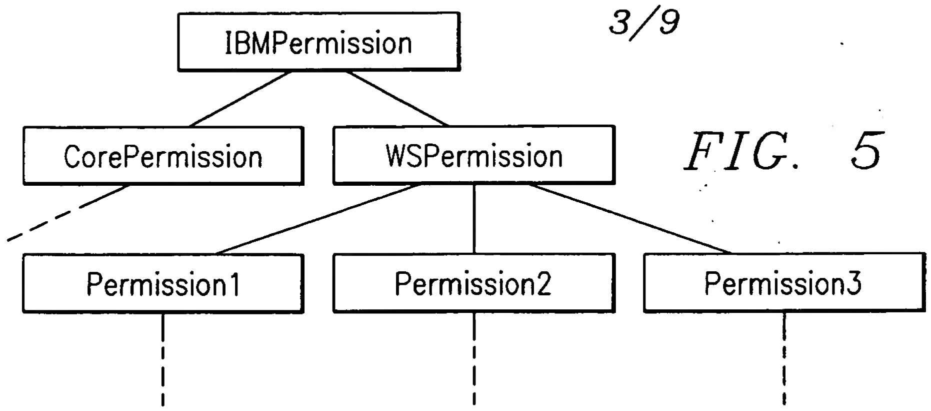

IBMPermission
CorePermission
WSPermission
Permission1
Permission2
Permission3

FIG. 5

FIG. 6

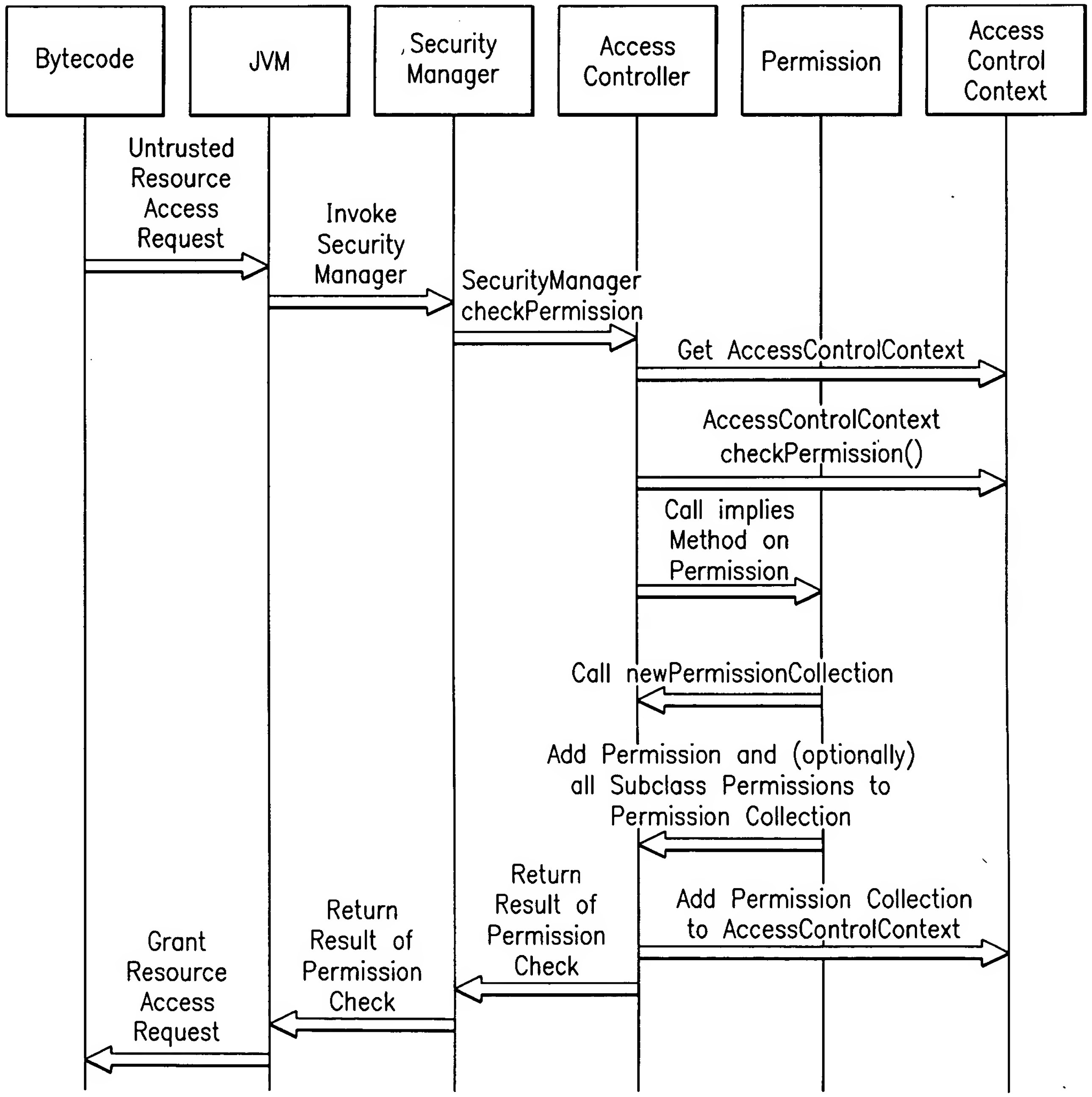

Bytecode
JVM
Security Manager
Access Controller
Permission
Access Control Context
Untrusted Resource Access Request
Invoke Security Manager
SecurityManager checkPermission
Get AccessControlContext
AccessControlContext checkPermission()
Call implies Method on Permission
Call newPermissionCollection
Add Permission and (optionally) all Subclass Permissions to Permission Collection
Add Permission Collection to AccessControlContext
Return Result of Permission Check
Return Result of Permission Check
Grant Resource Access Request

FIG. 7A

```
import java.security.BasicPermission;
import java.security.Permission;
import java.security.PermissionCollection;
import java.util.Hashtable;
import java.util.Enumeration;

public class IBMPermission extends BasicPermission
{
        public IBMPermission()
        {
                super (" ");
                System.out.println("Constructor IBMPermission() called");
        }
        public IBMPermission(String target)
        {
                super(target);
                System.out.println("Constructor IBMPermission(target) called");
        }

        public IBMPermission(String target, String actions)
        {
                super(target, actions);
                System.out.println("Constructor IBMPermission(target, actions) called");
        }
        public boolean implies(Permission perm)
        {
                System.out.println("IBMPermission.implies() called");

                if (perm instanceof IBMPermission)
                        return true;
                return false;
        }
        public PermissionCollection newPermissionCollection()
    {
                return new IBMPermissionCollection();
    }
}
```

FIG. 7B

```
final class IBMPermissionCollection extends PermissionCollection
      implements java.io.Serializable
{
      private Hashtable permissions;

      public IBMPermissionCollection()
      {
                 permissions = new Hashtable();
      }

      public void add(Permission permission)
      {
                 if (! (permission instanceof IBMPermission))
                    throw new IllegalArgumentException("Invalid Permission: " +
                                                                      permission);
                 IBMPermission ibmp = (IBMPermission) permission;
                 permissions.put(ibmp.getName(), permission);
      }

      public boolean implies (Permission permission)
      {
                 if (! (permission instanceof IBMPermission))
                       return false;

                 System.out.println("permission instanceof IBMPermission == true");

                 IBMPermission ibmp = (IBMPermission) permission;
                 String permName = ibmp.getName();
                 Permission x = (Permission) permissions.get(permName);

                 if (x != null)
                 {
                    System.out.println("We have a direct hit! " + x.getName());
                    return x.implies(permission);
                 }

                 Enumeration permEnum = permissions.elements();

                 while (permEnum.hasMoreElements())
                 {
                       x = (IBMPermission) permEnum.nextElement();
                       System.out.println(x.getName());

                       if (x.implies(permission))
                            return true;
                 }

                 return false;
         }

      public enumeration elements()
      {
                 return permissions.elements();
      }
}
```

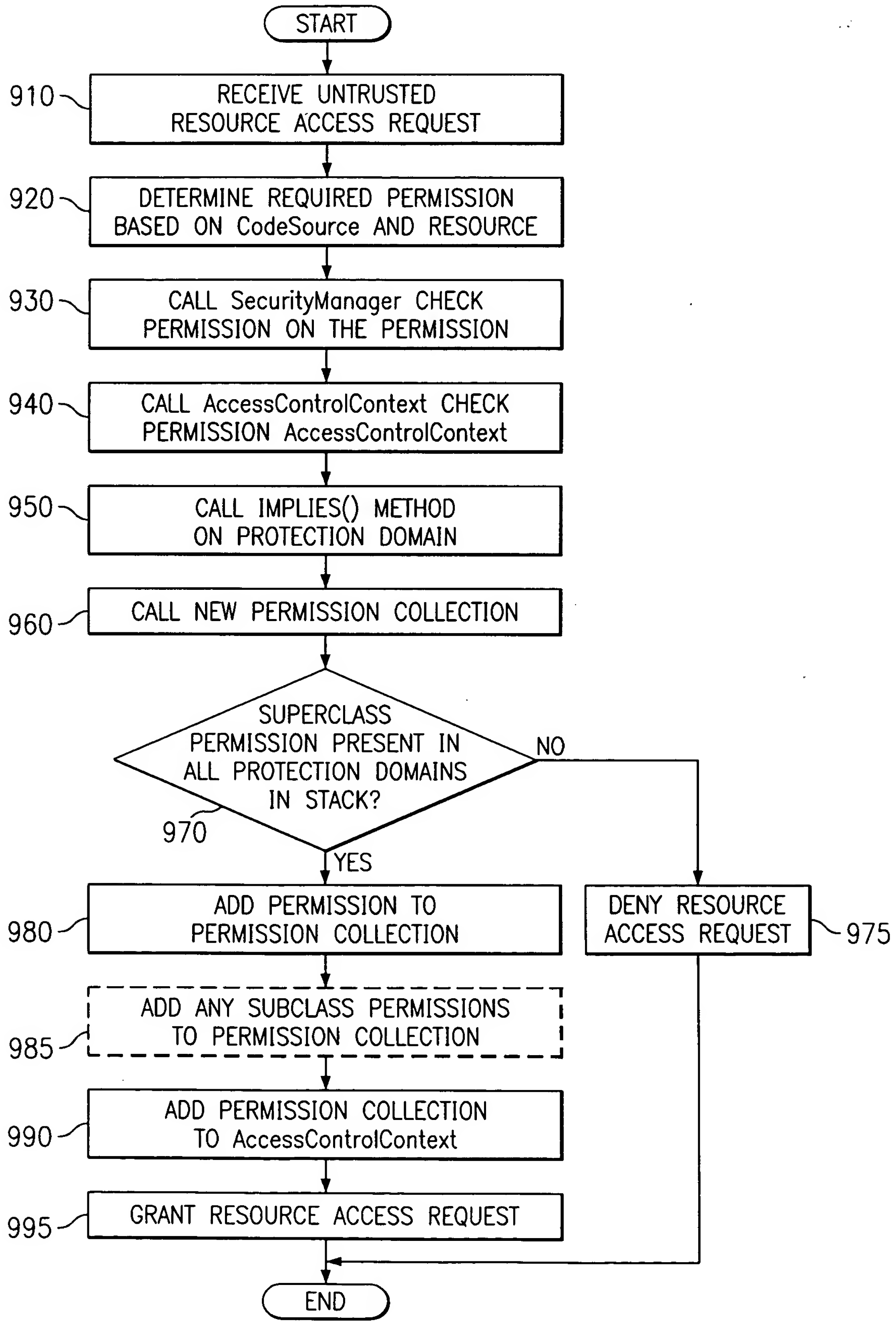
FIG. 7C

```
import java.security.PermissionCollection;
import java.security.AccessController;
import java.security.AccessControlContext;
import java.security.AccessControlException;

public class WSPermission extends IBMPermission
{
    public WSPermission(String target)
    {
        super(target);
        System.out.println("Constructor WSPermission(target) called");
    }
    public WSPermission(String target, String actions)
    {
        super(target, actions);
        System.out.println("Constructor WSPermission(target, actions) called");
    }
    public WSPermission()
    {
        super("");
        System.out.println("Constructor WSPermission() called");
    }

   /**
    * Returns a new IBMPermissionCollection object for storing IBMPermission
    * objects.
    * <p>
    * An IBMPermissionCollection stores a collection of
    * IBMPermission permissions.
    * <p>
    * IBMPermission objects must be stored in a manner that allows them
    * to be inserted in any order, but that also enables the
    * PermissionCollection <code>implies</code> method
    * to be implemented in an efficient (and consistent) manner.
    *
    * @return a new IBMPermissionCollection object suitable for
    *          storing IBMPermission's.
    */
     public PermissionCollection newPermissionCollection()
   {
        System.out.println("newPermissionCollection() was called");
        IBMPermissionCollection ibmPC = new IBMPermissionCollection();

        // the code here checks if an IBMPermissionCollection has been granted.
        // If yes, then the PermissionCollection returned by this
        // method should contain a WSPermission.

        AccessControlContext acc = AccessController.getContext();
        try
        {
            acc.checkPermission(new IBMPermission("PermissionTest"));
            ibmPC.add(new WSPermission("PermissionTest"));
        }
        catch (AccessControlException ace)
        {
            System.out.println("IBMPermission WAS NOT GRANTED");
        }
        return ibmPC;
    }
}
```

FIG. 8

```
import java.io.*;

public class PermissionTest
{
        public static void main(String args[])
        {
                try
                {
                        SecurityManager sm = System.getSecurityManager();

                        if (sm != null)
                        {
                                System.out.println("SecurityManager is checking for " +
                                                                    "WSPermission");

                              sm.checkPermission(new WSPermission("PermissionTest"));
                        }

                        System.out.println("WSPermission was granted. " +
                                                    "Permission testing
worked.\n\n\n");

                        File inputFile = new File("C:\\winzip.log");
                        FileInputStream fis = new FileInputStream(inputFile);
                        InputStreamReader isr = new InputStreamReader(fis);
                        BufferedReader br = new BufferedReader(isr);

                        String lineRead;
                        while ((lineRead = br.readLine()) != null)
                                System.out.println(lineRead);
                }

        catch (Exception e)
        {
                        e.printStackTrace();
        }
        }
}
```

FIG. 9


START
910
RECEIVE UNTRUSTED RESOURCE ACCESS REQUEST
920
DETERMINE REQUIRED PERMISSION BASED ON CodeSource AND RESOURCE
930
CALL SecurityManager CHECK PERMISSION ON THE PERMISSION
940
CALL AccessControlContext CHECK PERMISSION AccessControlContext
950
CALL IMPLIES() METHOD ON PROTECTION DOMAIN
960
CALL NEW PERMISSION COLLECTION
SUPERCLASS PERMISSION PRESENT IN ALL PROTECTION DOMAINS IN STACK?
970
NO
YES
980
ADD PERMISSION TO PERMISSION COLLECTION
DENY RESOURCE ACCESS REQUEST
975
985
ADD ANY SUBCLASS PERMISSIONS TO PERMISSION COLLECTION
990
ADD PERMISSION COLLECTION TO AccessControlContext
995
GRANT RESOURCE ACCESS REQUEST
END

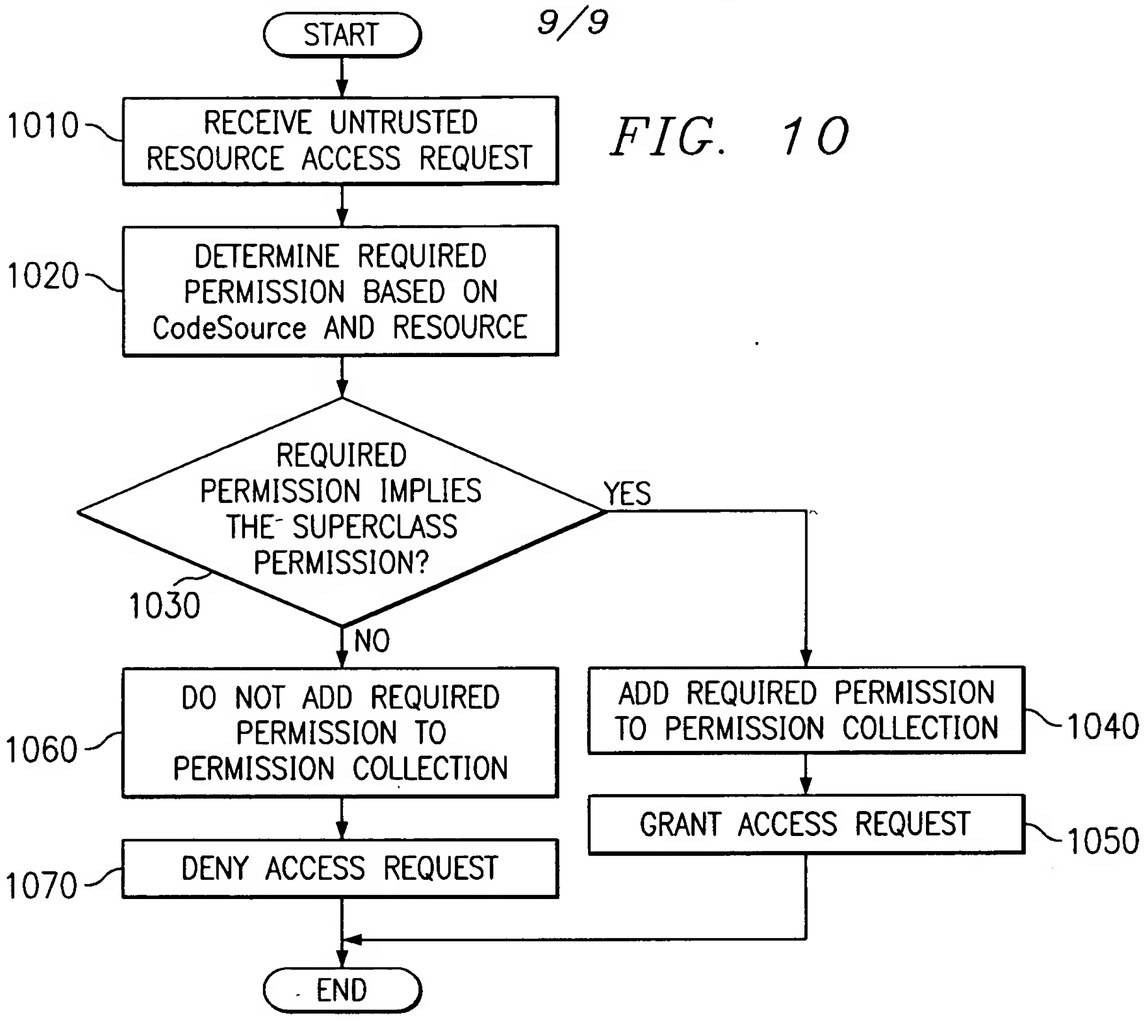

START
1010
RECEIVE UNTRUSTED RESOURCE ACCESS REQUEST
1020
DETERMINE REQUIRED PERMISSION BASED ON CodeSource AND RESOURCE
REQUIRED PERMISSION IMPLIES THE SUPERCLASS PERMISSION?
1030
YES
NO
ADD REQUIRED PERMISSION TO PERMISSION COLLECTION
1040
GRANT ACCESS REQUEST
1050
DO NOT ADD REQUIRED PERMISSION TO PERMISSION COLLECTION
1060
DENY ACCESS REQUEST
1070
END

FIG. 10

FIG. 11

```
package sun.security.provider;

import java.security.PermissionCollection;
import java.security.CodeSource;
import IBMPermission;
import WSPermission;

public class MarcoPolicy extends PolicyFile
{
        public PermissionCollection getPermissions(CodeSource codesource)
        {
                PermissionCollection pc = super.getPermissions(codesource);

                if (pc == null)
                        return null;

                if (pc.implies(new IBMPermission ("PermissionTest")))
                        pc.add(new WSPermission("PermissionTest"));

                return pc;
        }
}
```